Micriμm

Empowering Embedded Systems

μC/OS-II

μC/OS-View

μC/FS

μC/TCP-IP

μC/TFTPs

μC/HTTPs

and

the Fujitsu MB91403 (FR)

(Using the MB91943EB Evaluation Board)

Application Note

AN-9403

www.Micrium.com

Table of Contents

1.00	Introduction	3
1.01	Softune Workbench V6	4
1.02	Emulator	4
1.03	Required Files	5
1.04	Provided Files	7
2.00	Configuring μC/FS	9
2.01	Configuring μC/TCP-IP	11
2.02	Configuring the Example Application	12
3.00	`app.c`	13
3.01	Running the Example Application	14
References		18
Contacts		18

1.00 Introduction

The MB91943EB from Fujitsu is a robust development platform, offering a powerful microcontroller enhanced by numerous peripheral devices and memory resources. This microcontroller, the MB91403, is a versatile system-on-chip that features a 32-bit CPU based on Fujitsu's FR architecture. The CPU is connected to a variety of components, some of which reside on the surrounding board and others that are part of the MB91403 itself. One of the components belonging to this second group is the MB91403's Ethernet LAN controller, a device that allows your applications to efficiently perform an array of network tasks.

In order to expedite the process of developing network applications that utilize the MB91403's Ethernet LAN controller, you should consider using a TCP/IP protocol stack, such as Micrium's μC/TCP-IP. μC/TCP-IP is a reliable and easy-to-use TCP/IP module, and with the assistance afforded by this document, minimal effort is required to begin using the software on the MB91403. Likewise, μC/TCP-IP's optional add-on modules, such as μC/TFTPs and μC/HTTPs, are well-suited for use on the MB91403, so you can easily extend the capabilities of your network applications with these modules.

This document explains how to use μC/TCP-IP in conjunction with μC/TFTPs and μC/HTTPs on the MB91403. Advice for using Micrium's embedded file system, μC/FS, is also provided, since μC/TFTPs and μC/HTTPs both require the services of a file system. All of these modules, as well as Micrium's real-time operating system, μC/OS-II, and a runtime statistics module, μC/OS-View, are used in an example application that accompanies this document. While instructions for running this application are provided, the source code of the modules that make up the application is not. Because these modules must be licensed if they are used in a commercial product, you should contact Micrium for information on how to obtain them.

1.01 Softune Workbench V6

Softune Workbench is an integrated development environment that allows you to manage, build, and debug projects for Fujitsu processors. This tool was used to develop the sample application that accompanies this document. The version that was used (V60L3) actually doesn't recognize the MB91403 by default, so `911.csv`, the CPU information file provided with the software, had to be updated. If, similarly, the version of Softune that you will be using doesn't support the MB91403, you should contact Fujitsu for the appropriate updates.

1.02 Emulator

Although Softune Workbench offers a simulator debugger that allows you to test code without using your target hardware, the application that accompanies this document was tested on the previously described MB91943EB. Debugging with this board was facilitated by the MB2198-01 emulator. This emulator is shown in Figure 1-1.

Figure 1-1, MB2198-01 Emulator

1.03 Required Files

AN-9403.zip, the zip file that contains this document, also includes a Softune project that implements the example application cited in the previous sections. As Section 1.00 notes, this application utilizes several of Micrium's software modules, including μC/OS-II, μC/OS-View, μC/FS, μC/TCP-IP, μC/TFTPs, and μC/HTTPs. The source files for these modules must be obtained from Micrium, and the example project expects them to be located in the folders described below.

\Micrium\Software\uCOS-II\Source

The processor-independent files for μC/OS-II v2.81 or later should be placed in this folder. The appropriate files are easily identified, because, like the files comprising Micrium's other modules, they are normally distributed in folders named identically to those which are described in this section.

\Micrium\Software\uCOSView\Source

μC/OS-View's processor-independent files should be located in this folder. These files utilize the μC/OS-View port files provided with this document to transmit information about your application to a PC via the MB91403's UART.

\Micrium\Software\uCOSView\Viewer

This directory holds a Windows application capable of displaying statistics sent from an embedded target by μC/OS-View.

\Micrium\Software\uC-FS\FS\API

μC/FS is made up of several layers, and this folder contains the files that comprise the module's API layer.

\Micrium\Software\uC-FS\FS\CLIB

This directory contains files defining μC/FS versions of standard C library functions.

\Micrium\Software\uC-FS\FS\FSL

This folder holds the files that make up the file system layer of μC/FS. μC/FS allows you to use the FAT file system or EFS, a proprietary file system. The example application expects FAT to be used.

\Micrium\Software\uC-FS\FS\LBL

The files in this directory constitute the logical block layer of μC/FS.

\Micrium\Software\uC-FS\System\FS_X

This folder holds the files that define the interface between μC/FS and your application's operating system. The example application uses μC/OS-II.

\Micrium\Software\uC-FS\FS\DEVICE\ram

This directory contains the files that define the device driver used by µC/FS. The driver used by the example application allows files to be stored in the MB91943EB's SDRAM.

\Micrium\Software\uC-TCPIP\Source

µC/TCP-IP's device-independent files should be located in this directory.

\Micrium\Software\uC-TCPIP\NIC\ETHER\MB91403

This folder should be used to hold µC/TCP-IP's device driver for the MB91403's Ethernet LAN controller.

\Micrium\Software\uC-TCPIP\IF

µC/TCP-IP's network interface code is placed in this folder. A subfolder containing code corresponding to the specific interface being used also resides within this folder. The example application assumes that Ethernet will provide this interface.

\Micrium\Software\uC-TCPIP\OS\uCOS-II

The files that allow µC/TCP-IP to use services provided by µC/OS-II should be located in this folder.

\Micrium\Software\uC-TFTPs\Source

The source files for µC/TFTPs should be placed in this folder.

\Micrium\Software\uC-TFTPs\OS\uCOS-II

The files in this directory serve as the interface between µC/TFTPs and µC/OS-II.

\Micrium\Software\uC-HTTPs\Source

The source files for µC/HTTPs should be located in this folder.

\Micrium\Software\uC-HTTPs\OS\uCOS-II

This directory contains the files that provide the interface between µC/HTTPs and µC/OS-II

\Micrium\Software\uC-LIB

The files in this directory contain Micrium's implementations of the standard library routines used by µC/TCPIP. The routines are provided to simplify third-party certification of your products.

\Micrium\Software\uC-CPU

This folder contains files that define data types and configuration constants based on your CPU.

1.04 Provided Files

In addition to the source files covered by the previous section, the project provided with this document relies on several BSP and port files as a means of interfacing with the MB91943EB's peripherals. These files can be found in `AN-9403.zip`, and the example project expects the files to be located in the folders described below.

\Micrium\Software\uCOS-II\Ports\FR\Softune

This directory contains the processor-specific μC/OS-II files listed below. These port files are thoroughly described in AN-1015, an application note that can be downloaded from the Micrium Web site.

```
os_cpu_c.c
os_dbg.c
os_cpu.h
os_cpu_a.asm
os_cpu_i.asm
```

\Micrium\Software\uCOSView\Ports\FR\MB91403\Softune

This directory contains the μC/OS-View port files listed below. These files allow μC/OS-View to use the MB91403's UART to send data to a PC.

```
os_viewc.c
os_viewc.h
os_viewa.asm
```

\Micrium\Software\EvalBoards\Fujitsu\MB91943EB\Softune\OS_VIEW_FS_TCPIP_TFTP_HTTP\FS_Config

This folder contains `fs_conf.h`, which is used to configure μC/FS according to the needs of the example application.

\Micrium\Software\EvalBoards\Fujitsu\MB91943EB\Softune\BSP

The Board Support Package for the MB91943EB resides in this folder. These files declare routines to be used for initializing and otherwise manipulating peripheral devices.

\Micrium\Software\EvalBoards\Fujitsu\MB91943EB\Softune\Web_Pages

This folder contains a simple Web page. Using the example application you will be able to transfer this page to the MB91943EB via TFTP and then request the page with a Web browser.

\Micrium\Software\EvalBoards\Fujitsu\MB91943EB\Softune\OS_VIEW_FS_TCPIP_TFTP_HTTP

The example application is located in this directory. The application is composed of the files listed below.

```
app.c
includes.h
app_cfg.h
os_cfg.h
net_cfg.h
OS_VIEW_FS_TCPIP_TFTP_HTTP.*
```

`app.c` is the actual application, and `includes.h` is a master include file used by the application. `os_cfg.h`, `app_cfg.h`, and `net_cfg.h` are configuration files, and the prefix attached to each of these files indicates that file's purpose: `app_cfg.h` is used to configure your application, `os_cfg.h` configures μC/OS-II, and `net_cfg.h` is μC/TCP-IP's configuration file. `OS_VIEW_FS_TCPIP_TFTP_HTTP.*` denotes the Softune Workbench project files.

2.00 Configuring μC/FS

μC/FS is configured via `fs_conf.h`. There are numerous parameters that may be configured through `fs_conf.h`, and you should consult the μC/FS User's Manual for descriptions of the constants representing these parameters. The configuration file used by the example application only specifies values for a portion of these constants; the remaining configurable parameters are defined using the default values that μC/FS provides in `FS_ConfDefaults.h`. The constants for which default values are not used are described in Listing 2-1.

Listing 2-1, `fs_conf.h`

```
#ifndef  _FS_CONF_H_
#define  _FS_CONF_H_


#define  FS_USE_RAMDISK_DRIVER    1                                  /* (1) */

#define  FS_USE_FLASH_DRIVER      0
#define  FS_USE_GENDEV_DRIVER     0
#define  FS_USE_IDE_DRIVER        0
#define  FS_USE_MMC_DRIVER        0
#define  FS_USE_SMC_DRIVER        0

#define  FS_DEBUG_LEVEL           5                                  /* (2) */
#define  FS_MAXOPEN               3

#define  FS_OS_UCOS_II            1                                  /* (3) */

#define  FS_POSIX_DIR_SUPPORT     1
#define  FS_SUPPORT_FAT           1
#define  FS_SUPPORT_EFS           0

#if  FS_USE_RAMDISK_DRIVER

    #define  FS_RR_BLOCKNUM    1024                                  /* (4) */
    #define  FS_RR_BLOCKSIZE   0x200
    #define  FS_RAMDISK_ADDR   0x80080000

#endif

#endif
```

L2-1(1) FS_USE_RAMDISK_DRIVER indicates that the RAM driver will be used. This driver is selected so that the example application can store files on the MB91943EB's SDRAM.

L2-1(2) FS_DEBUG_LEVEL selects the types of messages that will be displayed by μC/FS. 5 is the highest value that can be specified for this constant, resulting in any messages generated by μC/FS being passed to `printf()`. `printf()` must actually be implemented, however, in order for these messages to be seen. More information on `printf()`, and the low-level, hardware-specific functions that it calls, can be found in the Softune C/C++ compiler documentation.

L2-1(3) If `FS_OS_UCOS_II` is set to `1`, μC/OS-II provides the operating system services needed when the file system is used in a multi-tasking environment.

L2-1(4) This group of constants further describes the portion of RAM that will be used by μC/FS. `FS_RR_BLOCKSIZE` indicates the block (or sector) size that will be used when μC/FS accesses RAM, and `FS_RR_BLOCKNUM` indicates the number of such blocks available. μC/FS assumes that these blocks of RAM begin at the address specified by `FS_RAMDISK_ADDR`.

2.01 Configuring μC/TCP-IP

Like μC/FS, μC/TCP-IP is configured by specifying values for constants that represent a multitude of configurable parameters. μC/TCP-IP does not, however, rely on a pair of configuration files, such as `fs_conf.h` and `FS_ConfDefaults.h`, with one file defining default configuration values and the other file overriding these definitions with values appropriate for a given application. Instead, μC/TCP-IP provides a single configuration file, `net_cfg.h`, that defines all of the module's configuration constants. You can directly modify the definitions in this file according to your application's needs.

Few of the constants in `net_cfg.h` need to be changed to accommodate the example application. `NET_NIC_CFG_INT_CTRL_EN`, which, as its name indicates, enables or disables support for an interrupt controller, is one of only two constants that need to be redefined. The other constant that should be redefined is `NET_SOCK_CFG_BLOCK_SEL`, which determines the socket blocking behavior exercised by μC/TCP-IP. `NET_SOCK_CFG_BLOCK_SEL` should be defined as `NET_SOCK_BLOCK_SEL_BLOCK`, and `NET_NIC_CFG_INT_CTRL_EN` should be defined as `DEF_ENABLED`. Please note that both of these constants are correctly defined in the version of `net_cfg.h` provided with the example application, so you only need to make changes if you are using the template `net_cfg.h` provided with μC/TCP-IP.

2.02 Configuring the Example Application

As Section 1.04 notes, the example application is configured via `app_cfg.h`. Constants defined in this file specify a priority and stack size for each of the application's tasks. `app_cfg.h` also defines the configuration constants used by μC/TFTPs and μC/HTTPs. The section of the file pertaining to these modules is further described in Listing 2-2.

Listing 2-2, `app_cfg.h`

```
#define  HTTPs_OS_CFG_TASK_NAME    "HTTP (Server)"

#define  HTTPs_CFG_FS_ROOT         ""                                    /* (1) */

#define  HTTPs_CFG_DEFAULT_FILE    "index.htm"                           /* (2) */

#define  HTTPs_CFG_IPPORT                      80

#define  HTTPs_CFG_MAX_ACCEPT_RETRY             3
#define  HTTPs_CFG_MAX_TX_RETRY                 3

#define  HTTPs_CFG_MAX_VAR_SIZE               255
#define  HTTPs_CFG_MAX_VAL_SIZE               255

#define  HTTPs_CFG_HTML_NOT_FOUND_MSG  "<HTML>\r\n" \
                                       "<BODY>\r\n" \
                                       "<HEAD><TITLE>SYSTEM ERROR</TITLE></HEAD>\r\n" \
                                       "<H1>NOT FOUND</H1>\r\n" \
                                       "The requested object does not exist on this \
                                       server. \r\n" \
                                       "</BODY>\r\n" \
                                       "</HTML>\r\n"

#define  TFTPs_OS_CFG_TASK_NAME    "TFTP (Server)"

#define  TFTPs_CFG_IPPORT                      69
```

L2-2(1) This constant indicates the folder in which μC/HTTPs searches for Web pages. Since the root folder is specified, all files stored using μC/FS will be visible via μC/HTTPs. This configuration is suitable for the sample application, but you may want to use a different folder when designing other applications, in order to prevent μC/HTTPs from accessing all of your files.

L2-2(2) `HTTPs_CFG_DEFAULT_FILE` specifies the file to be returned when a Web browser's request doesn't include a file name. Thus, the example application returns `index.htm` when such requests are made.

3.00 app.c

app.c, which contains main(), initializes each of the modules used by the example application. μC/OS-II is initialized first, and a single task is created to perform the remaining initializations. This task, rather than directly invoking the initialization functions associated with each module, calls AppNetInit() to set up μC/TCP-IP, μC/TFTPs, and μC/HTTPs. AppNetInit() designates both the MAC and IP addresses that will be used to communicate with the MB91943EB, and this function uses values of 00:00:00:14:00:00 and 10.10.1.64, respectively. If this configuration is not suitable for your network, you should consult Listing 3-1 for an overview of the changes that you will need to make.

Listing 3-1, app.c

```
static  void  AppNetInit (void)
{
    CPU_BOOLEAN  init;
    NET_IP_ADDR  ip;
    NET_IP_ADDR  msk;
    NET_ERR      err;


    NetUtil_ASCII_to_MAC(APP_MAC_ADDR_STR, &NetIF_MAC_Addr[0]);            /* (1) */
    err  = Net_Init();

    ip   = NetASCII_Str_to_IP(APP_IP_ADDR_STR_THIS_HOST, &err);
    msk  = NetASCII_Str_to_IP(APP_IP_ADDR_STR_NET_MASK, &err);
    NetIP_CfgAddrThisHost(ip, msk);                                        /* (2) */

    init = TFTPs_Init(OS_TICKS_PER_SEC);

    init = HTTPs_Init();
    (void)init;
}
```

L3-1(1) If you wish to change the board's MAC address to a value other than 00:00:00:14:00:00, you will need to modify APP_MAC_ADDR_STR, which is defined at the top of the file.

L3-1(2) The IP address that will be used by the MB91943EB is specified by APP_IP_ADDR_STR_THIS_HOST. This constant can be redefined at the top of the file.

3.01 Running the Example Application

After tailoring the example application and your configuration files according to the guidelines offered in the previous sections, you will be able to build the application. You can then run the resulting executable on the MB91943EB, provided that Softune's debugger has been initialized for use with your hardware. If the debugger has not yet been initialized, Softune will start a setup wizard when you attempt to begin debugging. The options that should be selected through this wizard vary, of course, depending on your hardware. If you are using the MB2198-01 emulator described in Section 1.02, you can follow the instructions provided in AN-1403, another application note from Micrium, for setting up the debugger. Otherwise, you will need to consult the appropriate documentation from Fujitsu.

Once you have set up the debugger, and the example application is running on your MB91943EB, the board's LEDs will begin to blink. This is a visual cue that you can begin sending and receiving data with μC/TCP-IP, since the LEDs are manipulated after this module has been initialized. At this time, then, you may want to attempt to "ping" the board from your PC by using a command similar to that which can be seen in Figure 3-1. Please note that the actual command that you use will differ from the example command if your board must use an IP address other than `10.10.1.64`. The procedure for modifying the IP address used by the board is described in Section 3.00.

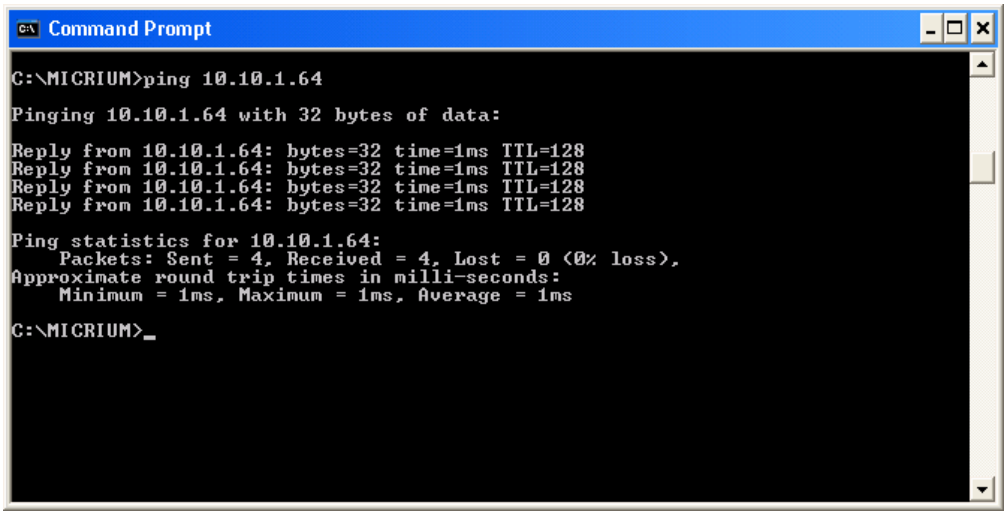
Command Prompt
C:\MICRIUM>ping 10.10.1.64
Pinging 10.10.1.64 with 32 bytes of data:
Reply from 10.10.1.64: bytes=32 time=1ms TTL=128
Reply from 10.10.1.64: bytes=32 time=1ms TTL=128
Reply from 10.10.1.64: bytes=32 time=1ms TTL=128
Reply from 10.10.1.64: bytes=32 time=1ms TTL=128
Ping statistics for 10.10.1.64:
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
Minimum = 1ms, Maximum = 1ms, Average = 1ms
C:\MICRIUM>_

Figure 3-1, Using `ping`

If you can successfully "ping" your board, then the example application is most likely configured appropriately for your network, and you can begin utilizing the services provided by μC/TCP-IP's add-on modules. μC/TFTPs, for example, can be used to copy files from your PC to the RAM disk instantiated on the board by μC/FS. Using a command similar to that which is shown in Figure 3-2, you can transfer the example Web page and image file included with this document to your board, allowing these files to later be viewed with a Web browser.

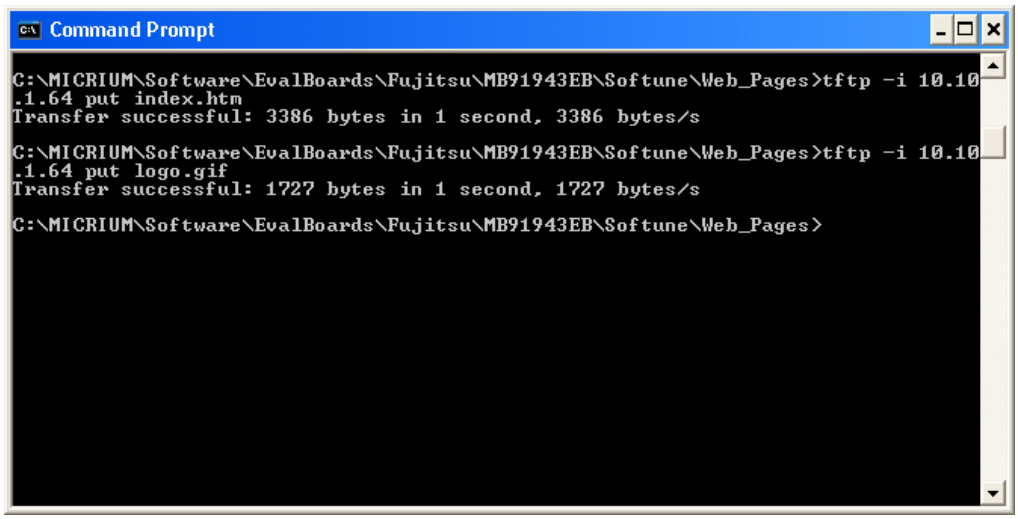
Command Prompt
C:\MICRIUM\Software\EvalBoards\Fujitsu\MB91943EB\Softune\Web_Pages>tftp -i 10.10
.1.64 put index.htm
Transfer successful: 3386 bytes in 1 second, 3386 bytes/s
C:\MICRIUM\Software\EvalBoards\Fujitsu\MB91943EB\Softune\Web_Pages>tftp -i 10.10
.1.64 put logo.gif
Transfer successful: 1727 bytes in 1 second, 1727 bytes/s
C:\MICRIUM\Software\EvalBoards\Fujitsu\MB91943EB\Softune\Web_Pages>

Figure 3-2, Transferring Files with `tftp`

After the provided Web page and image file have been placed on the board's RAM disk, μC/HTTPs will be able to access these files. Since `app_cfg.h` (the application configuration file described in Section 2.02) specifies `index.htm` as the default file provided by μC/HTTPs, you will be able to view this file by entering your board's IP address into a Web browser. Figure 3-3 demonstrates how `index.htm` can be accessed with Microsoft Internet Explorer if your board has an IP address of `10.10.1.64`.

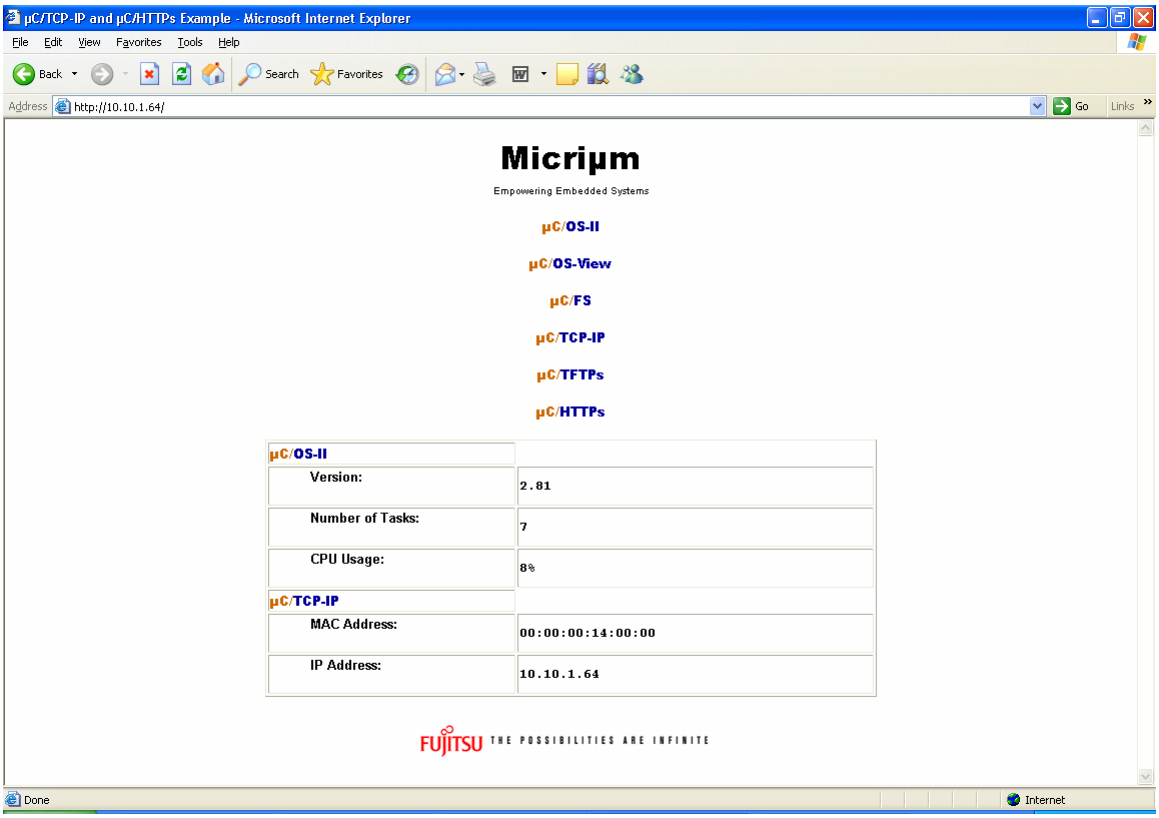

Figure 3-3, The Example Web Page

At any time while the example application is executing, you can start μC/OS-View's Windows application to view runtime statistics from the board. First, however, you must configure the Windows application through its `Setup` menu. In this menu, you will need to specify a baud rate of `19200`, and you should also designate the COM port to which the MB91943EB is connected. After this setup is complete, the Windows application should resemble the screenshot in Figure 3-4.

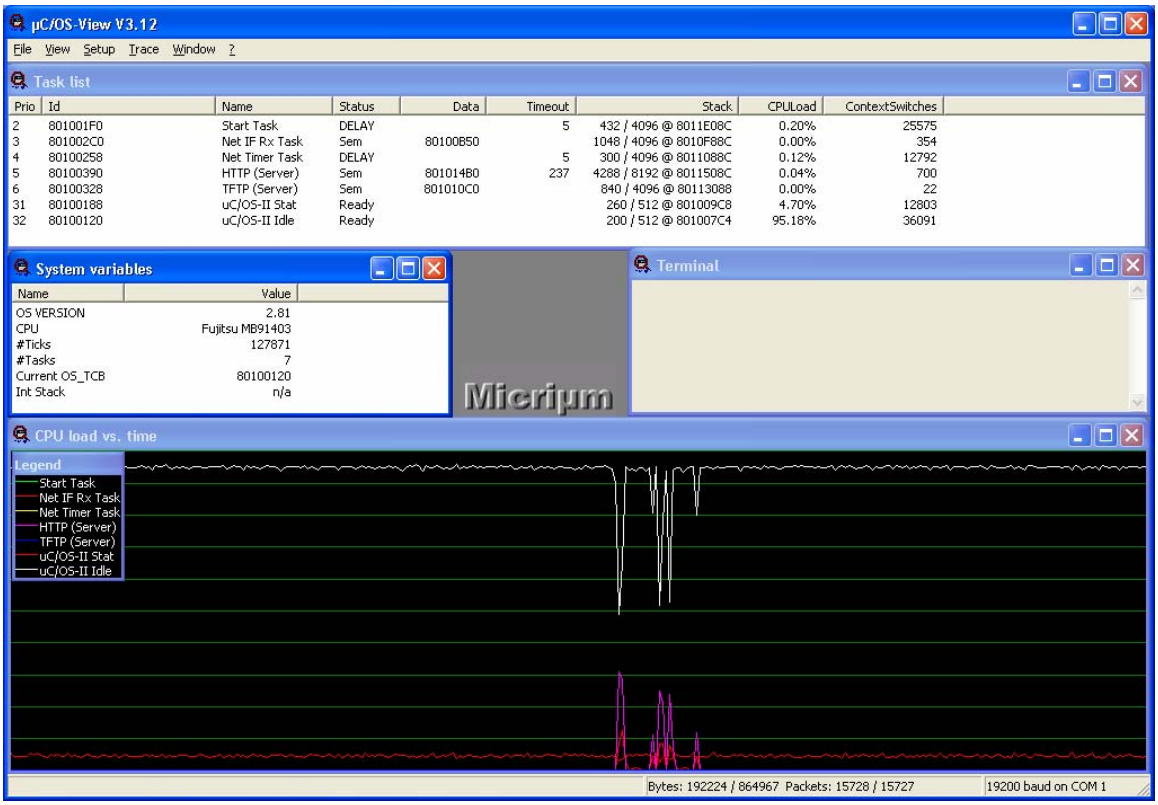
μC/OS-View V3.12
File View Setup Trace Window ?
Task list
Prio Id Name Status Data Timeout Stack CPULoad ContextSwitches
2 801001F0 Start Task DELAY 5 432 / 4096 @ 8011E08C 0.20% 25575
3 801002C0 Net IF Rx Task Sem 80100B50 1048 / 4096 @ 8010F88C 0.00% 354
4 80100258 Net Timer Task DELAY 5 300 / 4096 @ 8011088C 0.12% 12792
5 80100390 HTTP (Server) Sem 801014B0 237 4288 / 8192 @ 8011508C 0.04% 700
6 80100328 TFTP (Server) Sem 801010C0 840 / 4096 @ 80113088 0.00% 22
31 80100188 uC/OS-II Stat Ready 260 / 512 @ 801009C8 4.70% 12803
32 80100120 uC/OS-II Idle Ready 200 / 512 @ 801007C4 95.18% 36091
System variables
Name Value
OS VERSION 2.81
CPU Fujitsu MB91403
#Ticks 127871
#Tasks 7
Current OS_TCB 80100120
Int Stack n/a
Micriμm
Terminal
CPU load vs. time
Legend
Start Task
Net IF Rx Task
Net Timer Task
HTTP (Server)
TFTP (Server)
uC/OS-II Stat
uC/OS-II Idle
Bytes: 192224 / 864967 Packets: 15728 / 15727
19200 baud on COM 1

Figure 3-4, μC/OS-View

References

μC/OS-II***, The Real-Time Kernel, 2nd Edition***
Jean J. Labrosse
R&D Technical Books, 2002
ISBN 1-57820-103-9

Embedded Systems Building Blocks
Jean J. Labrosse
R&D Technical Books, 2000
ISBN 0-87930-604-1

Contacts

Micriμm
949 Crestview Circle
Weston, FL 33327
USA
+1 954 217 2036
+1 954 217 2037 (FAX)
e-mail: Jean.Labrosse@Micrium.com
WEB: www.Micrium.com

CMP Books
6600 Silacci Way
Gilroy, CA 95020 USA
Phone Orders: 1-800-500-6875
1-408-848-3854
Fax Orders: 1-408-848-5784
e-mail: cmp@rushorder.com
WEB: http://www.cmpbooks.com

Validated Software
Lafayette Business Park
2590 Trailridge Drive East, Suite 102
Lafayette, CO 80026
USA
+1 303 531 5290
+1 720 890 4700 (FAX)
e-mail: Sales@ValidatedSoftware.com
WEB: www.ValidatedSoftware.com